Computation Mechanism and Semantics for Logic Programs Based on Dataflow Networks

AUTHOR(S):

Yamasaki, Susumu

数理解析研究所講究録
655巻 1988年 124-146

データフローネットワークに基づく論理プログラムの計算機構と意味論

岡山大学工学部　　山崎　進

Computation Mechanism and Semantics for Logic Programs Based on Dataflow Networks

Susumu Yamasaki
Department of Information Technology
Okayama University, Tsushima, Okayama, Japan

Abstract. In this paper it is shown that the computation mechanism for a logic program is realized by a dataflow network. The network consists of two kinds of nodes and one way transmission channels connecting nodes. A node of the first kind is assigned to each Horn clause and a node of the second kind to each predicate symbol. The node of the first kind is interpreted as able to send a ground atom through an output channel after receiving a tuple of ground atoms from input channels, and denotes a relation among the sequences of ground atoms. The nondeterminism of the node, in choosing which tuple of ground atoms is to be received and which ground atom is to be sent, is eliminated and ensured fair by introducing oracles, for the network to be both sound and complete in finitely computing a logic program. The node of the second kind is a generalized fair merge operator, which provides a sequence of ground atoms to an output channel by fairly interleaving the sequences of input channels. It is necessary to realize so-called OR-nondeterminism essential in the deductions of logic programs, over sequence domains of ground atoms. The semantics of the network is given by fixpoint approach over sequence domains. It can be thought of as a semantics of the original logic program.

1. Introduction

There have been many discussions as to the semantics of logic programs since M.H. van Emden and R.A.Kowalski defined it from model-theoretic, fixpoint and operational approaches [1,2,3,12,15,23]. As far as the finite computation of logic programs is taken into account, the minimal Herbrand models of them are thought of as essential, as has already been made clear in [3,14,23]. Suppose that a set of Horn clauses is given as a logic program. Then, its minimal Herbrand model is regarded as denoting the set of all atoms derivable by finite deductions from it. Its minimal Herbrand model is also a least fixpoint of a function (from the power set of its Herbrand base to itself), which is associated with the logic program.

In this paper, it is shown that the semantics of logic programs can be defined over sequence domains from the Herbrand base. The motivation of the present paper is to study the relationship between the dataflow network and the computation mechanism for logic programs. A dataflow network is constructed as a computation mechanism for logic programs. The network will consist of nodes and channels connecting nodes. The channel transmits a sequence of ground atoms one way. The node receives sequences of ground atoms and sends a sequence of ground atoms. Mathematically, the denotations of channels are finite or infinite sequences of ground atoms. The nodes are thought of as defining relations among the sequences of ground atoms or functions of them. The relation or function each node denotes is in accordance with the behaviour of receiving and sending sequences of ground atoms.

The main concern is whether or not the denotation of a logic program, which is its minimal Herbrand model, can be expressed over sequence domains of ground atoms. Also we have a question of how the denotation of a logic program could be well-defined over sequence domains.

To answer the question, we have the first kind of nodes

each of which corresponds to a Horn clause in a given logic program, and is interpreted as sending ground atoms after receiving ground atoms, in accordance with the generation of ground atoms from the Horn clause and received ground atoms by bottom-up inferences. Then each node of the first kind will be defined to receive input sequences of ground atoms and send an output sequence. Next we have the second kind of nodes which are interpreted as 'fairly' merging input sequences of ground atoms and emitting an output sequence. Here it is meant by 'fairness' that there is no case where any part of any input sequence is neglected and does not appear in the output for ever. The emitted sequence is an input of some node of the first kind through a channel. On the other hand, the inputs of each node of the second kind emanate from the first kind of nodes. Each node of the second kind fairly interleaves more than one alternative sequences which can be an input of some node of the first kind. It is necessary to simulate OR-nondeterminism in the bottom-up inferences of Horn clauses, by means of sequences of ground atoms.

Thus, for a given logic program, we will have a network consisting of two kinds of nodes (which receive and send sequences of ground atoms), and channels transmitting sequences of ground atoms. It is different from a pure dataflow in [13], since (1) it contains the nodes realizing fair merging, and (2) each node of the first kind does not denote a function but a relation and has nondeterminism in deciding: (a) which ground atoms in sequences are to be received, and (b) which ground atom among possible atoms is to be sent out. We say that the network is complete if the ground atom is emitted by some channel whenever it is in the minimal Herbrand model of the given logic program. Also, the network is said sound if the ground atom is in the minimal Herbrand model whenever it is emitted by some channel. Note that the soundness and completeness of the network are the conditions to realize the computation mechanism of the originally given logic program. In order that

the network may be not only sound but also complete, it is sufficient for each node of the first kind to be fairly nondeterministic in deciding which ground atoms are to be received and which ground atom is to be sent.

We have a class of fair infinite sequences of natural numbers as oracles for each node of the first kind to be ensured fair in the above decisions and to be determinant. By the term: 'a fair infinite sequence of natural numbers', it is meant that any natural number occurs in the sequence an arbitrary number of times. We have a simple method to provide a class of such fair infinite sequences. Then, the whole network is reduced to a network consisting of nodes fairly interleaving sequences. The semantics of the network is defined by fixpoint approach, based on the denotations of channels over sequence domains of ground atoms. It can be regarded as a semantics of the originally given logic program.

2.Preliminaries

A logic program is a finite set of Horn clauses. A Horn clause is a clause of the form $A \leftarrow B_1 \ldots B_n$ $(n \geq 0)$, where A is either an atom or empty, and $B_1, \ldots, B_n$ are atoms. An atom is an expression of the form $P(t_1, \ldots, t_m)$ where P is a predicate symbol and $t_1, \ldots, t_m$ are terms.

A term is defined recursively: (i) a variable is a term; (ii) $f(t_1, \ldots, t_k)$ $(k \geq 0)$ is a term if f is an k-place function symbol and $t_1, \ldots, t_k$ are terms. (A 0-place function symbol is a constant symbol.)

A substitution is a finite set of the form $\{x_1 | t_1, \ldots, x_n | t_n\}$, where each x_i is a variable and each t_i is a term such that there is no occurrence of x_i in t_i. For a substitution σ and an atom A, $A\sigma$ is an atom obtained by substituting terms in σ for all the variables of σ occurring in A simultaneously.

Given a logic program S, the Herbrand universe Hu(S) of S is defined as follows:

$$H_0=\begin{cases}\{a \mid a \text{ is a constant symbol in } S\} \\ \quad \text{if } S \text{ contains at least one constant in } S, \\ \{b\} \quad (b: \text{some constant symbol}) \quad \text{otherwise,}\end{cases}$$

$$H_i=H_{i-1}\cup\{f(t_1,\ldots,t_n)\mid f: \text{a function symbol in } S,\ t_j\in H_{i-1}\},$$

$$Hu(S)=\bigcup_{i\in\omega} H_i,$$

where ω is the set of all natural numbers.

The Herbrand base H_S of S is

$$H_S=\{P(t_1,\ldots,t_m)\mid P: \text{a predicate symbol in } S,\ t_j\in Hu(S)\}.$$

Given a set F, #F means the cardinal number of F.

3.Network Constructed from Logic Program

3.1.Outline of Network

We shall have a method of transforming logic programs into networks consisting of nodes and one way channels. There are two kinds of nodes in a network. we assign a node of the first kind to each Horn clause $A\leftarrow B_1\ldots B_m$ ($m\geq 0$). The node is attached to an output channel, say, C and m input channels, say, $C_1,\ldots,C_m$. The node is interpreted as able to send out a ground atom $A\theta$ to the output channel C after receiving a tuple of ground atoms $B_1\theta,\ldots,B_m\theta$ from the input channels $C_1,\ldots,C_m$, respectively, where θ is a substitution. This assignment is in accordance with the generation of $A\theta\leftarrow$ from $\{A\leftarrow B_1\ldots B_m,\ B_1\theta\leftarrow,\ldots,\ B_m\theta\leftarrow\}$. In general, there are possibly more than one tuples of ground atoms to be received. Also, there is nondeterminism in sending a ground atom every time the node receives a tuple of ground atoms.

The output channel of a node for a Horn clause H_1 will be related to an input channel of another node for a Horn

clause H_2 if the procedure name (head) of H_1 has the same predicate symbol as that of some atom in the procedure body of H_2. Because the ground atoms sent out by the former node for H_1 can be received by the latter for H_2. There are possibly more than one output channels, say, $D_1,\ldots,D_p$, to be related to an input channel, say, D of the node for H_2. To get ground atoms emanating from the channels $D_1,\ldots,D_p$ through the input channel D, it is effective to have a node which has, as inputs, the relevant output channels $D_1,\ldots,D_p$ and, as an output, the input channel D, interleaving its input sequences from $D_1,\ldots,D_p$ to provide an output sequence to D. The interleaving is 'fair' in the sense that there is no case where any segment of any input sequence is neglected and does not appear in the output. Such a node is refered to as the second kind in the following context. Thus, the output channel of a node of the first kind will be an input channel of a node of the second kind and vice versa. Since all the atoms transmitted through $D_1,\ldots,D_p$ and D are regarded as having the same predicate symbol, it is seen that a node of the second kind should be defined in accordance with the predicate symbol.

The above network is a dataflow network and is regarded as a computation mechanism. The network is formally

$$Net=(N_C,\ N_M,\ C)$$

where N_C and N_M are sets of nodes such that they are disjoint and $C \subset N_C \times N_M \cup N_M \times N_C$ is a set of channels. Note that N_C is the set which consists of the nodes refered to as the first kind. N_M is the set of the nodes of the second kind.

We construct a network Net^S for a given logic program S, and define its semantics in order that Net^S may be a legal computation mechanism of S. To do so, several notations as to the logic program S are necessary.

$Atom^S$ denotes the set of atoms the forms of which appear in Horn clauses of S. $Pred^S$ means the set of predicate

symbols appearing in S.

(3.1) right: $S \to 2^{Atom^S}$ and left: $S \to Atom^S \cup \{\Phi\}$ are defined as follows:

$right(A \leftarrow B_1 \ldots B_m) = \{B_1, \ldots, B_m\}$ for $A \leftarrow B_1 \ldots B_m \in S$.

$$left(A \leftarrow B_1 \ldots B_m) = \begin{cases} A & \text{if } A \text{ is not empty,} \\ \Phi & \text{otherwise, for } A \leftarrow B_1 \ldots B_m \in S, \end{cases}$$

where Φ is a special symbol not in $Atom^S$ nor in $Pred^S$. (The purpose of using Φ will be explained later.)

(3.2) PRED: $Atom^S \cup \{\Phi\} \to Pred^S \cup \{\Phi\}$ is defined by

$$\begin{cases} PRED(\ P(t_1, \ldots, t_m)\ ) = P \text{ for } P(t_1, \ldots, t_m) \in Atom^S, \\ PRED(\ \Phi\ ) = \Phi. \end{cases}$$

Before formally defining the construction of a network $Net^S = (N_C, N_M, C)$ from a logic program S, we have an outline as to the connections among nodes.

Firstly, we shall have an illustrated node in N_C for a Horn clause $A \leftarrow B_1 \ldots B_m$ in S such that each C_i is an input channel from a node in N_M for $PRED(B_i)$ $(1 \le i \le m)$, and C_0 is an output channel to a node in N_M for PRED(A).

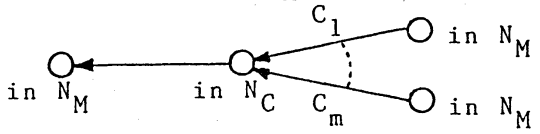

Fig.1 Connection of a node in N_C with channels

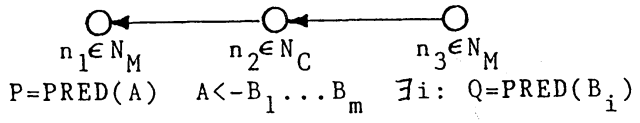

Fig.2 Connections among nodes

Secondly, we shall assign a node to each predicate symbol in $Pred^S$ or Φ. Φ is necessary to define a node in N_M, to which the nodes corresponding to goal statements (Horn clauses whose heads are empty) are connected. As illustrated in Fig. 2, if a node n_1 in N_M is defined for a predi-

cate symbol P or Φ, and a node n_2 in N_C is defined for a Horn clause of the form $A<-B_1 \ldots B_m$ such that PRED(A)=P, then the output channel of the node n_2 is an input channel of n_1. On the other hand, an output channel of a node n_3 is connected with a node n_2 in N_C, if the predicate symbol for n_3 is in an atom of the procedure body of a Horn clause for n_2. There may be more than one output channels of each node in N_M. But they are regarded as emanating from one output port of the node in N_M, since all the output channels denote the same sequence of ground atoms. There is no output channel of the node for Φ.

3.2.Formal Definition of Network

Formally, the construction of a network $Net^S=(N_C,N_M,C)$ from a logic program S is

(3.3) to define mappings in: $N_C->2^C$, out: $N_C->C$,

IN: $N_M->2^C$ and OUT: $N_M->2^C$, and

bijections Clause: $S->N_C$ and Merge: $Pred^S \cup \{\Phi\}->N_M$

such that

(1) for $H \in S$,

(i) $B \in$ right(H)
iff (Merge(PRED(B)),Clause(H)) $\in$ in(Clause(H)),

(ii) out(Clause(H))=(Clause(H),Merge(left(H))), and

(2) for $P \in Pred^S \cup \{\Phi\}$,

(i) P appears in right(H) for $H \in S$
iff (Merge(P),Clause(H)) $\in$ OUT(Merge(P)),

(ii) P=PRED(left(H)) for $H \in S$
iff (Clause(H),Merge(P)) $\in$ IN(Merge(P)).

Example 1 Let S be {N(0)<-, N(s(x))<-N(x) }, where N is a predicate symbol, 0 is a constant symbol, s is a function symbol and x is a variable.
Then N_M={ Merge(N) }, since there is only one predicate symbol N in S. N_C= {n_1,n_2} , where n_1= Clause(N(0)<-) and n_2=Clause(N(s(x))<-N(x)).

For N(0)<- in S, the condition (i) of (1) in (3.3) is satisfied, since right(N(0)<-) and $in(n_1)$ are empty. The condition (ii) of (1) is that $out(n_1)=(n_1,Merge(N))$.
For N(s(x))<-N(x) in S, (i) $N(x) \in N(s(x))$<-N(x) iff $(Merge(N),n_2) \in in(n_2)$, and (ii) $out(n_2)=(n_2,Merge(N))$.
For $N \in Pred^S$,

(i) N appears in right(N(s(x))<-N(x))
iff $(Merge(N),n_2) \in OUT(Merge(N))$, and

(ii) N=PRED(left(N(0)<-))
iff $(n_1,Merge(N)) \in IN(Merge(N))$ and
N=PRED(left(N(s(x))<-N(x)))
iff $(n_2,Merge(N)) \in IN(Merge(N))$.

Thus, $Net^S=(\{n_1,n_2\}, Merge(N), \{C_1,C_2,C_3\})$ is illustrated as follows:

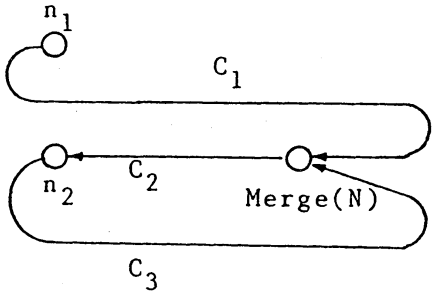

Fig.3 A network Net^S

3.3. Denotations of Channels

To consider the semantics of the network Net^S, we first have a domain $D=H_S \cup \{\Box\} \cup \{\tau\}$ for the denotations of channels. Intuitively speaking, τ denotes a time delay and is the same as the hiaton introduced in [20,24]. τ means that there is no output from the node in N_C even if it receives a tuple of ground atoms. $\Box$ represents that there exists no atom.

For D, D^∞ denotes the set of all finite and infinite sequences from D. $\perp$ is the empty sequence which is regarded as an element of D^∞. For $u \in D^\infty$ and $p \in \omega$, u(p) means the (p+1)-th element of u.

To get a sequence by eliminating τ from a given sequence in D^∞, we define a function E: $D^\infty \to (H_S \cup \{\square\})^\infty$ recursively:

(3.4) $E(\bot)=\bot$; $E(\tau u)=u$; $E(au)=aE(u)$ $(a\neq\tau)$.

Note that uv denotes a concatenation of u and v as sequences, for $u, v \in D^\infty$.

The denotations of channels are defined in D^∞.
Then a relation on $(D^\infty)^{\#\mathbf{in}(n)}$ and D^∞ is assigned to each n in N_C.

Assume that n=**Clause**(H) for $H=A\leftarrow B_1 \ldots B_m \in S$. Let the denotations of channels in **in**(n) be $u_{H1},\ldots,u_{Hm} \in D^\infty$, where #in(n)=m. Also let the denotation of out(n) be $v_H \in D^\infty$.

We define $v_H(p)$ for $p \in \omega$ as follows:
Intuitively speaking from the point of view of operational approach, the (p+1)-th symbol of v_H, sent out as an output of n, is in the set the member of which is the expression of the form $A\sigma \in H_S \cup \{\square\}$ if each $B_i\sigma$ matches (q_i+1)-th symbol of $E(u_{Hi})$ $(q_i \leq p)$, received through a corresponding input channel. It is assumed that the output sequence of length p depends on only the input sequences up to length p, for the simplicity of further treatments. To denote an m-tuple $(q_1,\ldots,q_m)$ by a natural number q which is larger than each element of the tuple, we define a bijection $I_m: \omega \to \omega^m$ such that if $I_m(p)=(p_1,\ldots,p_m)$ then $p_i \leq p$. On the other hand, to get the i-th element of an m-tuple, we define a projection $J_{m,i}: \omega^m \to \omega$ by $J_{m,i}(p_1,\ldots,p_m)=p_i$.

Formally, let

(3.5) $v_H(p) \in \bigcup_{q\leq p} Out_H(q)$
if the set in the right-hand side is not empty, and
$v_H(p)=\tau$ otherwise, for $p \in \omega$,

where

(3.6)

$Out_H(q)=\{A\sigma$ in $H_S \mid \exists\sigma$: $E(u_{Hi})(J_{m,i}(I_m(q)))=B_i\sigma,\ 1\leq i\leq m\}$
if A is not empty,

$Out_H(q)=\square$
if A is empty and
for some σ $E(u_{Hi})(J_{m,i}(I_m(q)))=B_i\sigma$, $1\leq i\leq m$, and

$Out_H(q)=\tau$ otherwise.

Thus, in (3.5), there is nondeterminism, in deciding $v_H(p)$ for $p\in\omega$, due to:

(i) which positions of input sequences are chosen for the matching with the right-hand side's atoms of H, and
(ii) which element is chosen as an output among the set $\bigcup_{q\leq p} Out_H(q)$, for the selected positions of inputs.

Also, a function $(D^\infty)^{\#IN(n)}\to D^\infty$ is assigned to $n\in N_M$ as follows:

As already mentioned, n has only one port from which the output channels emanate, and all the output channels are regarded as denoting the same sequence. We refer to the fair merge functions, which are based on the dmerge function in [19,20].

For any set K, K^∞ means the set of all finite and infinite sequences from K, containing the empty sequence $\perp_K$.
$\{0,1\}^\omega$ means the set of all infinite sequences from $\{0,1\}$.

$dmerge_K: K^\infty\times K^\infty\times\{0,1\}^\omega\to K^\infty$
is defined by

$$
\begin{aligned}
&dmerge_k(u,\perp_K,1\delta)=dmerge_K(\perp_K,u,0\delta)=\perp_K,\\
&dmerge_K(au,v,0\delta)=admerge_K(u,v,\delta) \text{ for } a\in K,\\
&dmerge_K(u,bv,1\delta)=bdmerge_K(u,v,\delta) \text{ for } b\in K.
\end{aligned}
\tag{3.7}
$$

(3.8) For $\delta\in(0^*011^*)^\omega$, $FM^{K,\delta}: K^\infty\times K^\infty\to K^\infty$ means a function such that $FM^{K,\delta}(u,v)=dmerge_K(u,v,\delta)$.

$FM^{K,\delta}$ is said a fair merge function. Note that $(0^*011^*)^\omega$

means the set of all infinite sequences from the set denoted by (0*011*). That is, each sequence of $(0*011*)^{\omega}$ contains unbounded occurrences of both 0 and 1. Thus $FM^{K,\delta}(u,v)$ provides a sequence w by interleaving u and v without neglecting any segment of them.

Let FM(p,α) (p≥2) mean a function of p inputs, which is defined by connecting some p-1 fair merge functions, as illustrated in Fig. 4, where $\alpha \in (\ 0(0*011*)\ )^{p-1}$ such that $\delta_i = J_{p-1,i}(\alpha)$.

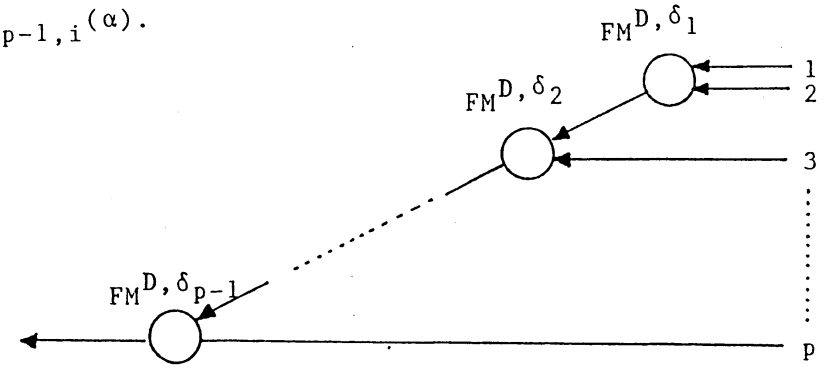

Fig.4 A function composed of p-1 fair merge functions

As easily seen, FM(p) provides a sequence by interleaving p input sequences fairly, that is, without neglecting any segment of any input sequence. The function denoted by FM(p) is called a generalized fair merge function.

Finally $FM(\#IN(n),\alpha_n)$ is assigned to the node n in N_M. That is, each node in N_M is interpreted as a generalized fair merge function.

3.4.Fair Nondeterminism in Network

The denotations of channels, satisfying the relations and functions assigned to the nodes, should guarantee the soundness and completeness of the network $\mathbf{Net}^S$ computing a given logic program S. Here soundness means that any ground atom provided by any output channel is a member of the minimal Herbrand model of S. Completeness means that any member of the minimal Herbrand model of S is emitted by some output channel.

For the soundness and completeness of $\mathbf{Net}^S$, nondeterminism in (3.5) is now eliminated and a function satisfying (3.5) will be assigned to each node in N_C.

We shall need fairness in such nondeterminism. It means that any positions of input sequences are chosen for the matchings of right-hand side's atoms in the Horn clause, in a finite time, and any element able to be sent out is really emitted in a finite time. The fairness requires that any m-tuple of positions is selected an arbitrary number of times if one of the input sequences is infinite. Accordingly, any possible member is emitted an arbitrary number of times. To indicate an m-tuple of the positions of m input sequences, it is sufficient to have only natural number if we use the bijection I_m. Next we want an infinite sequence in which any natural number appears an arbitrary number of times, in order to denote the sequence of fairly selected m-tuples of positions concerning input sequences for each node in N_C.

Definition 1 We say that a sequence in ω^ω is fair if any $p \in \omega$ occurs an arbitrary number of times in the sequence.

Note that ω^ω means the set of all infinite sequences from ω.

To get a class of fair sequences in ω^ω, we construct a simple network by the way as illustrated below, where $FM^{\omega, 0\delta}$ is defined for $0\delta \in (0*011*)^\omega$ by (3.8) and $T:\omega^\infty \to \omega^\infty$ is an operator defined recursively: $T(\perp_\omega) = \perp_\omega$; $T(au)=s(a)T(u)$ $(a \in \omega, u \in \omega^\infty)$ for the successor function s.

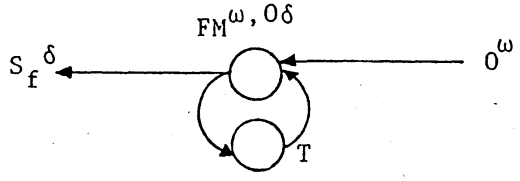

Fig.5 A network to provide fair infinite sequence

Let S_f^δ be the sequence generated by the above network, that is, by the recursion equation: $S_f^\delta = FM^{\omega, 0\delta}(0^\omega, T(S_f^\delta))$.

Proposition 1 Let $\delta \in (0*011*)^\omega$. Then

(1) S_f^δ is fair.

(2) $S_f^\delta(j) \leq j$ for $j \in \omega$.

Proof (1) $S_f{}^{\delta}=0u$ for some $u\in\omega^{\infty}$, since $S_f{}^{\delta}(0)$ comes from the input 0^{ω} through $FM^{\omega,0\delta}$. Now let Occur(k,h) mean that k occurs, in u, h times.

(i) It is seen that Occur(0,h) for any $h\in\omega$, because 0^{ω} is an input of $FM^{\omega,0\delta}$ and 0 appears in its output an arbitrary number of times.

(ii) Assume that for some h, Occur(n,h) for $n\leq k$. Then, since k+1 should be an output of T, and at the same time an input of $FM^{\omega,0\delta}$, Occur(k+1,h). By mathematical induction, for some $h\in\omega$, Occur(k,h) for any k.

(iii) Since Occur(0,1) from (i), it follows from (ii) that Occur(n,1) for any $n\in\omega$. Suppose that Occur(k,h) for any k. Since k should be an output of T, k-1 should be an input of T, that is, k-1 occurs, in u, as many times as k. Thus Occur(k-1,h). Finally Occur(0,h). It follows from (i) that Occur(0,h+1). Therefore, there is no case that Occur(k,h+1) does not hold. This completes the induction step. That is, Occur(k,h) for any k and any h.

(2) If j=0, then the proposition holds, since $S_f{}^{\delta}=0u$. Assume that $S_f{}^{\delta}(h)\leq h$ for $h\leq k$. Since $S_f{}^{\delta}(k)+1$ or 0 can be the input of $FM^{\omega,0\delta}$, $S_f{}^{\delta}(k+1)\leq k+1$. This completes the induction step.

Q.E.D.

Observing (3.5), (3.6) and Proposition 1, we see that $S_f{}^{\delta}(p)\leq p$ is used to denote a natural number in accordance with an m-tuple of positions of input sequences for the decision of $v_H(p)$. For each natural number q corresponding to an m-tuple of positions, the set $Out_H(q)$, whose member can be emitted to the output channel, is determined by (3.6).

To have a correspondence of such a natural number q with $Out_H(q)$ and to enumerate the members in $Out_H(q)$, we define

(3.9) a function $R_H: \omega \to (2^{(H_S)} \to \omega)$

such that $R_H(q)$ is a injection from $Out_H(q)$ to ω, and $R_H(q)(A)<\#Out_H(q)$ for A in $Out_H(q)$.

Now assume that for $p_0<p_1<\ldots\in\omega$ $S_f^{\delta}(p_0)=S_f^{\delta}(p_1)=\ldots$. Then $Out_H(S_f^{\delta}(p_0))=Out_H(S_f^{\delta}(p_1))=\ldots$. To provide any member in $Out_H(S_f^{\delta}(p_0))$, it is sufficient to enumerate its members by $R_H(\ S_f^{\delta}(p_0)\ )^{-1}: \omega \to Out_H(\ S_f^{\delta}(p_0)\ )$ and by $S_f^{\gamma}(0)$, $S_f^{\gamma}(1),\ldots$, on the basis of some fair sequence S_f^{γ}.

To get $i\in\omega$ from $S_f^{\delta}\in\omega^{\omega}$ and $S_f^{\delta}(p_i)$, we define

$$Ord: (\omega^{\omega})\to(\omega\to\omega)$$

by

$$(3.10)\quad Ord(S_f^{\delta})(q)=\#\{r|\ S_f^{\delta}(r)=S_f^{\delta}(q)\}-1.$$

Then, $R_H(\ S_f^{\delta}(p_i)\ )^{-1}(S_f^{\gamma}(\ Ord(S_f^{\delta})(p_i)))$
$=R_H(\ S_f^{\delta}(p_i)\ )^{-1}(\ S_f^{\gamma}(i)\ )$.

Finally, by eliminating nondeterminism in (3.5), a function to be assigned to the node **Clause**(H) is defined by

$$(3.11)\quad v_H(p)=R_H(\ S_f^{\delta}(p)\ )^{-1}(S_f^{\gamma}(Ord(S_f^{\delta})(p\ )))$$
if the right-hand side is defined, and
$v_H(p)=\tau$ otherwise, for $p\in\omega$,

where δ and γ are in $(0*011*)^{\omega}$.

From now on, given a network Net^S for a logic program S, it is assumed that the function is defined by (3.11) to each node in N_C and a generalized fair merge function is assigned to each node in N_M.

4.Soundness and Completeness of Network for Logic Program

In this section, we show that the network Net^S is a legal computation mechanism in the sense that it is both sound and complete.

Definition 2 Let H_S be the Herbrand base of a logic program S. $T_S: 2^{(H_S)} \to 2^{(H_S)}$ is defined by

$T_S(I)=\{A\sigma \in H_S \mid \exists \sigma: A \leftarrow B_1 \ldots B_k \in S, \{B_1\sigma, \ldots, B_k\sigma\} \subset I\}$

for $I \in 2^{(H_S)}$. Let $Sem(S)=\bigcap\{I \mid T_S(I) \subset I\}$.

Note that Sem(S) is the minimal Herbrand model [23]. We have the formal definition of soundness and completeness in the following.

Definition 3 Assume the network $Net^S=(N_C,N_M,C)$ for a logic program $S=\{H_1,\ldots,H_k\}$. Net^S is sound if $left(H_i)\theta$ (θ: substitution), which appears in $out(Clause(H_i))$, is in Sem(S). Net^S is complete if $left(H_i)\theta \in Sem(S)$ (θ: substitution) necessarily appears in $out(Clause(H_i))$.
(For left, see (3.1). For Clause and out, see (3.3).)

We have the following proposition, which states the soundness and completeness of the network Net^S.

Proposition 2 Given a logic program $S=\{H_1,\ldots,H_k\}$ and Net^S, $left(H_i)\theta \in Sem(S)$ iff $left(H_i)\theta$ appears in the channel $out(Clause(H_i))$, for a substitution θ.

Proof Note that $Sem(S)=\bigcup_{j\in\omega} T_S{}^j(\phi)$ (ϕ: the empty set) [3,23].

(=>: completeness) It is proved, by mathematical induction on j such that $left(H_i)\theta \in T_S{}^j(\phi)$, that $left(H_i)\theta$ appears in $out(Clause(H_i))$.

(i) Assume that $left(H_i)\theta \in T_S{}^1(\phi) \subset Sem(S)$. Then there exists a Horn clause H_i of the form $A\leftarrow$. Thus, if $left(H_i)\theta \in H_S$, then $left(H_i)\theta \in Out_{H_i}(q)$ for any q. It means that $left(H_i)\theta$ appears in $out(Clause(H_i))$, since the fairness in the choice of outputs is guaranteed.

(ii) Assume that any element of $T_S{}^h(\phi) \subset Sem(S)$ appears in $out(Clause(H_i))$ for some H_i if $h \leq j$. Now let $left(H_i)\theta \in T_S{}^{j+1}(\phi)$. Also suppose that $H_i=left(H_i)\leftarrow B_1 \ldots B_{m_i}$. Then, $B_1\psi,\ldots,B_{m_i}\psi \in T_S{}^h(\phi)$ ($h \leq j$) for some substitution ψ such

that left(H_i)θ =(left(H_i)ψ)σ for some substitution σ. By the induction hypothesis, $B_1\psi,\ldots,B_{m_i}\psi$ are emitted by output channels. Therefore, they are all in input channels of Clause(H_i) through the nodes in N_M. Since the tuple $(B_1\psi,\ldots,B_{m_i}\psi)$ of ground atoms is necessarily selected as a received tuple by 'fairness', left(H_i)θ should be provided by Clause(H_i), due to 'fairness' in deciding the outputs. This completes the induction step.

(<=: soundness) (i) Assume that H_i=left(H_i)<-. By (3.11), any output of Clause(H_i) takes the form left(H_i) $\theta \in H_S$ for a substitution θ. Thus, left(H_i)$\theta \in$ Sem(S).

(ii) Assume that H_i=left(H_i)<-$B_1\ldots B_{m_i}$ (m_i>0) and that any ground atoms appearing in the input channels of the node Clause(H_i) are in Sem(S). If left(H_i)θ appears in out(Clause(H_i)), then there exist $D_1,\ldots,D_{m_i}$ in the input channels of Clause(H_i) such that $D_j=B_j\theta$ for $1\leq j\leq m_i$. By the definition of Sem(S), if $D_1,\ldots,D_{m_i}$ are in Sem(S), then left(H_i)θ is in Sem(S).

(iii) Any ground atom, appearing in an input channel of any node in N_C, has appeared in the output channel of some node in N_C via a node in N_M. It follows from (i) and (ii) that left(H_i)θ is in Sem(S), as far as it appears in the output channel of Clause(H_i).

Q.E.D.

5. Semantics of Network for Logic Program

In the previous section, it was seen that the network Net^S is both sound and complete. It means that the network is a legal finite-computation mechanism and its semantics can be regarded as a semantics of the original logic program S.

A fixpoint semantics of the network Net^S is well-defined. We shall see this. Suppose $Net^S=(N_C,N_M,C)$ and $C=\{C_1,\ldots,C_k\}$. The denotations of $C_1,\ldots,$ and C_k are assumed to be $u_1,\ldots,$

and u_k, respectively.

Let E be the function defined in (3.4). we say a function $f: (D^{\infty})^m \to D^{\infty}$ is asynchronous if $f(E(v_1),\ldots,E(v_m))= f(v_1,\ldots,v_m)$, that is, if f is insensitive to occurrences of τ. It is evident from (3.6) and (3.11) that the function assined to each node in N_C is asynchronous.

A partial order $\sqsubset$ on D^{∞} is defined by:

$u \sqsubset v$ for u, v in D^{∞} iff $v=uw$ for some w in D^{∞}.

It is extended to act on $(D^{\infty})^m$ $(m \geq 1)$:

$(u_1,\ldots,u_m) \sqsubset (v_1,\ldots,v_m)$ iff $u_p \sqsubset v_p$ for $1 \leq p \leq m$.

The least upper bound of a set $F \subset (D^{\infty})^m$ is the element u such that

(i) $v \sqsubset u$ for any v in F, and
(ii) if $v \sqsubset w$ for any v in F, then $u \sqsubset w$.

The least upper bound of F is denoted by $\sqcup F$. We see that any chain $\{ v_0 \sqsubset v_1 \sqsubset \ldots \sqsubset v_n \sqsubset \ldots\}$ over $(D^{\infty})^m$ has a least upper bound.

Definition 4 $f: (D^{\infty})^m \to D^{\infty}$ is continuous iff, for any chain $\{ v_0 \sqsubset v_1 \sqsubset \cdots \; v_n \sqsubset \ldots\}$, $f(\sqcup\{ v_i \mid i \in \omega\}) = \sqcup\{ f(v_i) \mid i \in \omega \}$.

Proposition 3 Assume that we have the network $Net^S = (N_C, N_M, \{ C_1,\ldots,C_k \})$, where the denotation of C_i is u_i $(1 \leq i \leq k)$. Then, for i=1,...,k, there exists either a generalized fair merge function or an asynchronous, continuous function $f_i: (D^{\infty})^k \to D^{\infty}$ such that $u_i = f_i(u_1,\ldots,u_k)$.

Proof It is sufficient for the proof to show that each

node in N_C is continuous. (Note that the function assigned to each node in N_C is asynchronous, as already mentioned. Obviously each node in N_C denotes a generalized fair merge function.) In (3.9), $R_H(\ S_f\ (p)\ )^{-1}$ is a surjection from ω to $Out_H(\ S_f\ (p)\ )$, which is the set depending on $E(u_{H1}),\ldots,E(u_{Hm})$ for the node Clause(H). By Proposition 1, $S_f{}^{\delta}(p)\leq p$. Thus, $v_H(p)$ is a function of the tuple obtained by truncating the denotation of each input channel up to length p+1. Therefore we could put $u_i=f_i(u_1,\ldots,u_k)$ for f_i: $(D^{\infty})^k\to D^{\infty}$, which is concerned with the node whose output channel has the denotation u_i. Now let $v[p]$ be $v(0)v(1)\ldots v(p)$ for $v\in D^{\infty}$.

Since $f_i(u_1[p],\ldots,u_k[p])\sqsubset f_i(u_1,\ldots,u_k)$ for any $p\in\omega$,

$$\sqcup\{f_i(u_1[p],\ldots,u_k[p])\mid p\in\omega\}\sqsubset f_i(u_1,\ldots,u_k)$$
$$=f_i(\sqcup\{(u_1[p],..,u_k[p])\mid p\in\omega\}).$$

On the other hand, for any $p\in\omega$,

$$u_i[p]=f_i(u_1,\ldots,u_k)[p]\sqsubset f_i(u_1[p],\ldots,u_k[p]),$$

since $u_i(p)$ is determined by $u_1[q],\ldots,u_k[q]$ for some $q\leq p$, based on (3.11). Thus,

$$f_i(\sqcup\{(u_1[p],\ldots,u_k[p])\mid p\in\omega\})=f_i(u_1,\ldots,u_k)$$
$$=\sqcup\{f_i(u_1,\ldots,u_k)[p]\mid p\in\omega\}$$
$$\sqsubset\sqcup\{f_i(u_1[p],\ldots,u_k[p])\mid p\in\omega\}.$$

This completes the continuity of f_i.

Q.E.D.

Proposition 4 $FM^{D,\delta}$ is continuous.

Proof By (3.8), $FM^{D,\delta}(u,v)=dmerge(u,v,\delta)$. Note that $dmerge(\perp,v,0\delta_1)=dmerge(u,\perp,1\delta_2)=\perp$. Therefore, for any chain $\{(u_0,v_0)\sqsubset(u_1,v_1)\sqsubset\ldots\}$, whose least upper bound is (u,v), $\sqcup\{FM^{D,\delta}(u_i,v_i)\}\sqsubset FM^{D,\delta}(u,v)=FM^{D,\delta}(\sqcup\{(u_i,v_i)\mid i\in\omega\})$.

On the other hand, because of the fairness of $FM^{D,\delta}$, for any $p\in\omega$ there exists i such that $FM^{D,\delta}(u,v)[p]\sqsubset FM^{D,\delta}(u_i,v_i)$. Therefore,

$$FM^{D,\delta}(\sqcup\{(u_i,v_i)\mid i\in\omega\})=FM^{D,\delta}(u,v)$$
$$=\sqcup\{FM^{D,\delta}(u,v)[p]\mid p\in\omega\}$$
$$\sqsubset\sqcup\{FM^{D,\delta}(u_i,v_i)\mid i\in\omega\}.$$

This completes the continuity of $FM^{D,\delta}$. Q.E.D.

It follows from Propositions 3 and 4 that for a network $Net^S=(N_C,N_M,\{ C_1,\ldots,C_k \})$ there is a continuous function $f_S: (D^\infty)^k \to (D^\infty)^k$ such that each u_i is the denotation of C_i and $f_S(u_1,\ldots,u_k)=(f_1(u_1,\ldots,u_k),\ldots,f_k(u_1,\ldots,u_k))$. By the elementary theory as to continuous functions, there exists a least fixpoint $fixf_S$ of f_S. Indeed, it is not hard to see that $fixf_S= \bigcup \{ f_S^{\,i}(\bot,\ldots,\bot) \mid i \in \omega \}$, where $f_S^{\,i}(\bot,\ldots,\bot)$ is defined recursively:

$$f_S^{\,0}(\bot,\ldots,\bot)=(\bot,\ldots,\bot),$$
$$f_S^{\,i}(\bot,\ldots,\bot)=f_S(\, f_S^{\,i-1}(\bot,\ldots,\bot)\,).$$

For the least fixpoint $fixf_S=(U_1,\ldots,U_k)$, $(E(U_1),\ldots,E(U_k))$ can be regarded as a behaviour (semantics) of the network Net^S. Because of the soundness and completeness of Net^S computing S, it is also interpreted as a semantics of S, which is defined using the sequence domain $(H_S \cup \{\Box\})$.

6.Concluding Remarks

In this paper, a dataflow network was constructed as a computation mechanism for a given logic program. It consists of nodes and channels. The channel transmits and denotes a sequence of ground atoms. The node receives and sends sequences of ground atoms, denoting a relation among them or a generalized fair merge function. Although each node in accordance with a Horn clause denotes a relation and has nondeterminism in receiving and sending ground atoms, the nondeterminism can be eliminated by introducing oracles based on fair infinite sequences of natural numbers in order that the network may be both sound and complete as a computation mechanism. Here it is meant by soundness and completeness that any member is in the minimal Herbrand model of the original logic program iff it is emitted by some channel in the network. Then the whole network is regarded as a continuous function of a tuple of denotations

of channels. Eliminating time delay in the least fixpoint of the continuous function, we have a semantics of the network, which can be a semantics of the original logic program. This is the primary result of the paper.

The network of this paper is expected to be a basis on which we can perform the translation of logic programs into some functional language programs, say, dataflow language programs (in [4]).

As another aspect, it is shown that the computation of the constructed network corresponds to the "all-solutions" mode of Prolog over sequence domains, since the network is both sound and complete in computing a given logic program.

In this paper, the network deals with only sequences of ground atoms. We have a significant problem of how the network should be established in order that goal statements containing variables and answer substitutions can be computed. In [25], we have dealt with semantics of a Horn sentence, denoting the set of substitutions with which atoms are derivable by unit deduction from the Horn sentence, to get a direct correspondence between the semantics of the Horn sentence and the answer set concerned with its computation. Combining the treatment in that paper with the primary result of this paper, we can have a method to solve the problem, although it seems very awkward.

Acknowledgement

This was inspired by Professor Park's work. The author is greatly indebted to Professor Park for the suggestions on fair nondeterminism during the visits to University of Warwick and University of Edinburgh, the United Kingdom.

References

1. Abdallah,M.A.Nait, On the Interpretation of Infinite Computations in Logic Programming, Lecture Notes in Computer Science, 172: 358-370 (1984).
2. Abdallah,M.A.Nait, On Some Topological Properties of Logic Programs, Lecture Notes in Computer Science, 199: 310-319 (1985).
3. Apt,K.R. and van Emden,M.H., Contributions to the Theory of Logic Programming, J. ACM, 29: 841-864 (1982).
4. Ashcroft,E.A. and Wadge,W.W., Lucid-A Formal Systems for Writing and Proving Programs, SIAM J. Computing, 5: 336-354 (1976).
5. Ashcroft,E.A. and Wadge,W.W., Some Common Misconceptions about Lucid, Theory of Computation Report, No.32, Dept. of Computer Science, University of Warwick (1979).
6. Brookes,S.D., Hoare,C.A.R. and Roscoe,A.W., A Theory of Communicating Sequential Processes, J. ACM, 31: 560-597 (1984).
7. Clark,K. and Gregory,S., PARLOG: A Parallel Programming Language, Research Report DOC 83/5, Dept. of Computing, Imperial College (1983).
8. Clark,K. and Gregory,S., PARLOG: Parallel Programming in Logic, Research Report DOC 84/4, Dept. of Computing, Imperial College (1984).
9. De Nicola,R., Two Complete Axiom Systems for a Theory of Communicating Sequential Processes, Information and Control, 64: 136-172 (1985).
10. Faustini,A.A., An Operational Semantics for Pure Dataflow, Theory of Computation Report, No.38, Dept. of Computer Science, University of Warwick (1981).
11. Faustini,A.A., The Equivalence of an Operational and a Denotational Semantics for Pure Dataflow, Theory of Computation Report, No.41, Dept. of Computer Science, University of Warwick (1982).
12. Frauden,G., Logic Programming and Substitutions, Lecture

Notes in Computer Science, 199: 146-158 (1985).

13. Kahn,G., The Semantics of a Simple Language for Parallel Programming, Proc. IFIP 74: 471-475 (1974).
14. Lassez,J.L. and Maher,M.J., Closures and Fairness in the Semantics of Programming Logic, Theoretical Computer Science, 29: 167-184 (1984).
15. Lassez,J.L. and Maher,M.J., Maximal fixpoints of Logic Programs, Theoretical Computer Science, 39: 15-25 (1985).
16. Park,D., Fixpoint Induction and Proofs of Program Properties, Machine Intelligence, 5: 59-78 (1969).
17. Park,D., Finiteness Is Mu-Ineffable, Theory of Computation Report, No.3, Dept. of Computer Science, University of Warwick (1974).
18. Park,D., On the Semantics of Fair Parallelism, Lecture Notes in Computer Science, 86: 504-526 (1980).
19. Park,D., Concurrency and Automata on Infinite Sequences, Lecture Notes in Computer Science, 104: 167-183 (1981).
20. Park,D., The "Fairness" Problem and Nondeterministic Computing Networks, in: Foundations of Computer Science IV (de Bakker and van Leeuwen, eds.), Mathematisch Centrum, Amsterdam, 133-161 (1983).
21. Plotkin,G.D., A Powerdomain Construction, SIAM J. Computing, 5: 454-487 (1976).
22. Smyth,M.B., Powerdomains, JCSS, 16: 23-36 (1978).
23. Van Emden,M.H. and Kowalski,R.A., The Semantics of Predicate Logic as a Programming Language, J. ACM, 23: 733-742 (1982).
24. Wadge,W.W., An Extentional Treatment of Dataflow Deadlock, Lecture Notes in Computer Science, 70: 285-299 (1979).
25. Yamasaki,S., et al., A Fixpoint Semantics of Horn Sentences Based on Substitution Sets, to appear in Theoretical Computer Science (1987).